IN THE CLAIMS:

Please find below a listing of all of the pending claims. The statuses of the claims are set forth in parentheses.

1. (Original) A code verification system, comprising:

memory for storing a compiled program; and

a code verifier configured to analyze instructions of said program and to generate a plurality of type signatures based on said instructions, each of said type signatures indicating each input type constraint and each output type description for a respective one of said instructions, wherein said code verifier is configured to detect a type error by analyzing said type signatures.

2. (Original) The system of claim 1, wherein said code verifier is configured to compose one of said type signatures with another of said type signatures to form a single composed signature, said code verifier further configured to make a determination as to whether an input type constraint of said one type signature is acceptable to an output type description of said other type signature, said code verifier further configured to detect said type error based on said determination.

3. (Original) The system of claim 1, wherein said code verifier is further configured to analyze said program and to group instructions of said program into a plurality of code blocks, wherein said code verifier is configured to translate said code blocks into type signature blocks, each of said type signature blocks having one or more type signatures, said

code verifier further configured to compose the type signatures of each of said type signature blocks into a single respective composed type signature.

4. (Original) The system of claim 3, wherein said code verifier, in composing one of said type signature Mocks, is configured to compose a first type signature with a second type signature to form a single composed signature, said first and second type signatures included within said one type signature block, said code verifier further configured to make a determination as to whether an input type constraint of said first type signature is acceptable to an output type description of said second type signature, said code verifier further configured to detect said type error based on said determination.

5. (Original) A code verification system, comprising:

memory for storing a compiled program; and

a code verifier configured to analyze a code block of said program and to translate instructions within said code block into a plurality of type signatures, said code verifier further configured to compose said type signatures into a single composed type signature and to detect a type error by analyzing said type signatures.

6. (Original) The system of claim 5, wherein one of said type signatures includes a type description indicative of a type of an input consumed by one of said instructions when said one instruction is executed, wherein another of said type signatures includes a type description indicative of a type of an output produced by another of said instruction when

said other instruction is executed, and wherein said code verifier is further configured to detect said type error by comparing said type descriptions.

7. (Original) The system of claim 5, wherein said code verifier is further configured to analyze said program and to group instructions of said program into a plurality of code blocks, wherein said code verifier is configured to translate said code blocks into type signature blocks, each of said type signature blocks having one or more type signatures, said code verifier further configured to compose the type signatures of each of said type signature blocks into a single respective composed type signature.

8. (Original) A code verification method, comprising the steps of:

storing a compiled program;

generating a plurality of type signatures based on instructions within said program, each of said type signatures indicating each input type constraint and each output type description for a respective one of said instructions;

analyzing said type signatures; and

detecting a type error based on said analyzing step.

9. (Original) The method of claim 8, further comprising the steps of:

composing one of said type signatures with another of said type signatures thereby forming a single composed signature; and

determining whether an input type constraint of said one type signature is acceptable

to an output type description of said other type signature,

wherein said detecting step is further based on said determining step.

10. (Original) The method of claim 8, further comprising the steps of:

grouping instructions of said program into a plurality of code blocks;

translating said code blocks into type signature blocks, each of said type signature blocks having one or more type signatures; and

composing the type signatures of each said type signature blocks into a single respective composed type signature.

11. (Original) A code verification method, comprising the steps of:

storing a compiled program, said compiled program having at least one code block that includes a plurality of instructions;

translating said instructions into a plurality of type signatures;

composing said type signatures into a single composed type signature;

analyzing said type signatures; and

detecting a type error based on said analyzing step.

12. (Original) The method of claim 11, wherein one of said type signatures includes a type description indicative of a type of an input consumed by one of said instructions when said one instruction is executed, wherein another of said type signatures includes a type description indicative of a type of an output produced by another of said instructions when

said other instruction is executed, and wherein said analyzing step includes the step of comparing said type descriptions.

13. (Original) The method of claim 11, further comprising the steps of:

grouping instructions of said program into a plurality of code blocks;

translating said code blocks into type signature blocks, each of said type signature blocks having one or more type signatures; and

composing the type signatures of each of said signature blocks into a single respective composed type signature.

14. (New) A code verifier comprising:

means for generating a plurality of type signatures based on instructions within a compiled program, each of said type signatures indicating each input type constraint and each output type description for a respective one of said instructions;

means for analyzing said type signatures; and

means for detecting a type error based on said analyzing step.

15. (New) The code verifier of claim 14, further comprising:

means for composing one of said type signatures with another of said type signatures thereby forming a single composed signature; and

means for determining whether an input type constraint of said one type signature is acceptable to an output type description of said other type signature.

16. (New) The code verifier of claim 14, further comprising:

means for grouping instructions of said program into a plurality of code blocks;

means for translating said code blocks into type signature blocks, each of said type signature blocks having one or more type signatures; and

means for composing the type signatures of each said type signature blocks into a single respective composed type signature.